REMARKS

Posture of the case

Claims 1-16 were originally filed. This amendment is in response to Examiner's first Office action mailed June 5, 2007.

Non-Prior art objections and rejections

Specification

The disclosure stands objected to because there appeared to be a minor error in page 3, line 31 : "sae" in "the sae of variables" needs to be corrected to "state." Applicant has herein above responsively amended the paragraph.

Applicant also herein amends paragraph 0033 of the original application, as published, to correct an inadvertently omitted word that is obvious from the context.

Drawings

FIG. 4 (sheet 4/5) submitted with the filed application stands objected to on grounds that it did not include the label "470" mentioned in the specification, page 13. Applicant responsively herein above amends the *specification* to conform the description of the FIG. 4 in the specification to originally submitted FIG. 4.

35 U.S.C. 112

Claim 13 stands rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. In particular, the Office action holds it is unclear what is the object of "destination address." For examination Interpretation: the destination address.

Prior art rejections

Claims 1-16 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Hawley et al. (US Patent 5,533,192) hereafter Hawley in view of Stallman et al. ("Debugging with GDB," Free Software Foundation, 211 999) hereafter Stallman.

The present application describes aspects of at least one embodiment of the present invention that are clearly not suggested by the cited art. Applicant herein amends

and cancels claims and herein submits new claims as set out above in order to point out these patentable distinctions and overcome the rejection.

In particular, the present application describes the following aspects of one or more embodiments of the present invention that are clearly not suggested by the cited art:

> The techniques described herein provide a programmer with the facility of *inputting the desired flow of execution* for the program being debugged which, to the debugger, is its child program. The debugger detects the location where the flow is to be altered. The debugger changes the instruction pointer of the child program to achieve the desired change in execution flow. No instructions need be lost in the child program.
>
> In summary, the debugger alters the child program by inserting jump statements. The jump destination addresses are specified by a user. One or more instructions are replaced by a jump instruction. Therefore, this technique is used when the debugger determines that loss of a current instruction does not pose a problem.

Present application, as published, paragraph 0009 (emphasis added).

> *The process of changing the instruction pointer is automated.* Recompilation is avoided. *Automatically* changing the instruction pointer is desirable as an alternative to manually altering the instruction pointer for code that is "hit" frequently. After taking input from a user, the debugger inserts special breakpoints called "jump points". The jump points can be stored in a normal breakpoint list, or as a separate list. The contents of the jump point are listed below.
>
> (i) The location at which the jump is to be performed, namely the address of the relevant instruction.
>
> (ii) The destination location, which is the intended address at which execution should resume.
>
> (iii) The memory address that stores the instruction that is replaced while inserting a jump point.

Present application, as published, paragraph 0033 (emphasis added).

> *Changes to an existing debugger* are as follows. A user-interface for the debugger *provides a user with an option to specify how the flow is to be altered. The debugger is modified to save* the instruction at the point from where the jump is to be made. The debugger inserts exception-throwing instructions at locations from where the jump is to be made, and distinguishes these exceptions as jump point exceptions [rather] than that of normal breakpoints. Finally, the debugger alters the instruction point to the destination address specified by the user.

Present application, as published, paragraph 0046 (emphasis added).

In order to more particularly point out these novel and nonobvious features of the present invention, claim 1 is herein amended to point out that the inventive method is for *automatically* altering execution flow of a first computer program while the first program is executing (emphasis added), which is different than manually inserting breakpoints and different than what is taught or suggested in the cited art.

Further, amended claim 1 goes on to recite specific features of the method to further distinguish the invention. In particular, the inventive method includes the second

program "receiving . . . a desired flow of execution for the first program from a user predefined list, wherein the list includes a plurality of jump instructions, each jump instruction defining respective originating and destination source code locations in the first program," as amended claim 1 now states. The cited art does not teach or suggest this.

Further, amended claim 1 goes on to recite still further specific features of the method to distinguish the invention. In particular, the inventive method also includes "inserting the plurality of jump instructions into memory locations of the computer corresponding to respective originating source code locations of the first program defined by the respective jump instructions, wherein for each of the plurality of inserted jump instructions, execution flow in the first program jumps from the memory location corresponding to the respective jump instruction's originating source code location to the memory location corresponding to the respective jump instruction's defined destination source code location responsive to the first program encountering the memory location corresponding to the respective jump instruction's originating source code location, wherein the inserting is performed automatically by the second program responsive to the user predefined list," as amended claim 1 now states. The cited art does not teach or suggest this.

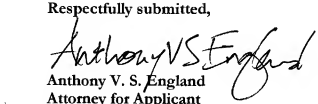
No new matter is presented, since the original application provides support for the amendments. See portions of the original specification set out herein above. See also Tables 2-4 and discussion thereof in the original specification for examples of inserting jump instructions into source code.

For these reasons, Applicant submits that claim 1 is allowable. Claims 15 and 16 are similarly amended herein, so that Applicant submits that these claims are also allowable for these reasons. Applicant submits that the claims 2, 4, 7-9, 12 and 17-27 are all allowable at least because they each depend on an allowable independent claim.

REQUESTED ACTION

Applicant submits that the invention as claimed in accordance with amendments submitted herein is patentably distinct, and hereby requests that Examiner grant allowance and prompt passage of the application to issuance.

Respectfully submitted,



Anthony V. S. England
Attorney for Applicant
Registration No. 35,129
512-477-7165
a@aengland.com